EE 321 – HW 2

Sum an array elements

- Consider a list L

  Where L = {2,11,7,50,4}

- In C

```
int sum=0;
for(int i=0; i<5; i++)
    sum += L[i];
```

- What about ASM?

VerySimpleCPU’s instructions

- ADD -> unsigned Add
- ADDi -> unsigned Add immediate
- NAND -> bitwise NAND
- NANDi -> bitwise NAND immediate
- SRL -> Shift Right if the shift amount (*B) is less than 32, otherwise Shift Left
- SRLi -> Shift Right if the shift amount (B) is less than 32, otherwise Shift Left
- LT -> if *A is Less Than *B then *A is set to 1, otherwise to 0.
- LTi -> if *A is Less Than B then *A is set to 1, otherwise to 0.
- CP -> Copy *B to *A
- CPi -> Copy B to *A
- **CPI -> (regular) Copy Indirect: Copy **B to *A**
- **CPIi -> (immediate) Copy Indirect: Copy * B to **A**
- BZJ -> Branch on Zero
- BZJi -> Jump (unconditional branch)

Representation of a list L in C

- Consider an array of N elements, nothing fancy:
  - N consecutive memory locations
  - L points to nothing more than the address of the 1st element
  - L[M] is just another way of writing *(L+M)
- Example:
  - Consider L starts at 1000
  - L[1] = *(1000 + 1) = *1001
  - In the inverse, *(1000 + 11) is L[11]
  - Etc.
- No overflow detection, the programmer has to take care of the boundary conditions

Accessing a list element

- Consider we are trying to read the 3rd element of L = {2,11,7,50,4}
- We are actually to do *100 = *(L+3) where L=1000
- This can be directly done with CP 100 1003. What about if the offset is i and we try to access *(L+i) ?
- CPI will help to solve this problem
- CPI instruction has the following behaviour:
  - *A <- *(*B)
  - It means
    - Go to address B and fetch the value written there that I will call C.
    - Now go to address C and retrieve the value there
- Let's start with well known offset access
  - CPi 104 1003		// *104 is 1003, the address of *(L+3)
  - CPI 100 104		// *100 has now the value *104 (3rd element in L)
- Let's follow the 2nd example with a variable offset i where i is in *101
  - CPi 104 1000		// *104 is 1000
  - ADD 104 101		// *104 has now 1000 + i as value
  - CPI 100 104		// *100 has now the element i's value

Writing to a list

- Consider the same example where L=1000 and

L = {2,11,7,50,4}.

This time, let's write 5 to an element of the array.

- Write to 3rd element

```
CPi 104 1003          // *104 = 1003
CPi 102 5             // *102 = 5
CPIi 104 102          // Copy 5 to 1003, i.e. L[3] = 5
```

- Write to ith element

```
CPi 104 1000          // *104 = 1000
ADD 104 101           // *104 = 1000 + i (i is in 101)
CPi 102 5             // *102 = 5
CPIi 104 102          // Copy 5 to 1000+i, i.e. L[i] = 5
```